UNIVERSITÉ DE NANTES

Travail d'étude et de recherche

Optimisation du Clustered Capacitated Vehicle Routing Problem

Encadrant :
M. Marc Sevaux

Auteurs :
Mohamed Cissé
Takfarinas Saber

7 mai 2012

Table des matières

Introduction

Le problème de tournées de véhicules est un problème d'optimisation combinatoire en nombres entiers qui a été introduit pour la première fois par Dantzig et Ramser en 1959 sous le nom de "The Truck Dispatching Problem".

Les deux chercheurs l'avaient introduit pour décrire un problème de livraison d'essence. Il est plus connu aujourd'hui sous le nom de Vehicle Routing Problem. C'est sans aucun doute l'un des problèmes les plus importants de transport et logistique. Il s'agit de satisfaire la demande d'un certain nombre de clients situés à proximité d'un dépôt.

Plusieurs variantes de ce problème existent dont notamment :

- le CVRP (Capaciatted Vehicle Routing Problem), où l'on rajoute, une notion de capacité limite pour l'ensemble véhicules de la flotte.
- le VRPTW (Vehicle Routing Problem with Time Windows) où la livraison pour chaque client doit être dans un intervalle fixé au préalable.
- le MDVRP (Multiple Depot Vehicle Routing Problem) où la flotte est répartie sur plusieurs dépôts. Ainsi, un consommateur peut être livré par des dépôts distincts.

Le problème de tournées de véhicules par groupes ou Clustered Vehicle Routing Problem est une variante du problème de tournées de véhicules dans laquelle les clients sont partagés en groupes. Une nouvelle contrainte apparaît donc lors de la résolution du problème. L'arrêt de la visite d'un groupe de clients n'intervient que si tous les clients de ce groupe ont été desservis.

De nombreuses emtreprises rencontrent ce type de problème. La majorité d'entre elles travaillent dans le secteur de la livraison de produits. De fait, le système actuel référence les clients grâce à leurs codes postaux. Ceci implique par conséquent, le transport de produits à destination d'une même ville par le même véhicule.

Parmi ces entreprises ont trouvent notamment TNT dont une étude lui a été consacrée dans [ref.1].

1 Le problème de tournées de véhicules

Grâce à une flotte de véhicules, l'objectif ici sera de visiter l'ensemble des clients. On s'aperçoit tout de suite que cet objectif correspond à la minimisation du temps de trajet ou de la distance parcourue par chaque véhicule.

Le problème de tournées de véhicules est généralement modélisé grâce aux acquis de la théorie des graphes. Soit $G = (S, A)$ un graphe complet dans lequel $S = \{0 \dots n\}$ représente un ensemble de sommets et A un ensemble d'arcs. On attribue communément au sommet 0 le rôle de dépôt. Donc, l'ensemble $S \backslash \{0\}$ correspond à l'ensemble des clients. On associe à chaque arc $(i, j) \in A$ un coût $d_{i,j}$ et représente le coût du trajet entre les sommets i et j.

Par convention, nous interdisons le trajet d'un véhicule d'un sommet i vers lui-même. On affecte ainsi au coût $d_{i,i}$ la valeur $+\infty$. Ce problème a été modélisé dans [ref.2] de la manière suivante :

- Q : capacité des véhicules
- n : le nombre de clients, dépôt exclu
- m : le nombre de véhicules
- $d_{i,j}$: coût (distance) du trajet du sommet i vers le sommet j, ce coût est non négatif.
- $x_{i,j,k}$: variable binaire valant 1 si l'arc (i, j) est parcouru par le véhicule k et 0 sinon

$$min \quad z = \sum_{i=1}^{n} \sum_{j=1}^{n} (d_{i,j}. \sum_{i=1}^{m} x_{i,j,k})$$

s.c.

$$\sum_{i=1}^{n} \sum_{k=1}^{m} x_{i,j,k} = 1, \quad \forall j \in \{1 \dots n\} \tag{1}$$

$$\sum_{j=1}^{n} \sum_{k=1}^{m} x_{i,j,k} = 1, \quad \forall i \in \{1 \dots n\} \tag{2}$$

$$\sum_{i=1}^{n} \sum_{l=1}^{n} x_{i,l,k} - \sum_{l=1}^{n} \sum_{j=1}^{n} x_{l,j,k} = 0, \quad \forall k \in \{1 \dots m\} \tag{3}$$

$$\sum_{j=1}^{n} x_{0,j,k} = 1, \quad \forall k \in \{1 \dots m\} \tag{4}$$

$$\sum_{i=1}^{n} x_{i,0,k} = 1, \quad \forall k \in \{1 \dots m\} \tag{5}$$

$$x_{i,j,k} \in \{0, 1\}, \quad \forall i, j \in \{1 \dots n\}, \quad \forall k \in \{1 \dots m\}$$

Les contraintes 1 et 2 encadrent le nombre d'arcs entrants et sortants de chaque sommet du graphe. En effet, le degré entrant et sortant de chaque sommet du graphe devra être égal à 1. Ces deux contraintes ne s'appliquent pas pour le dépôt car nous devons créer plusieurs cycles dont les extrémités sont le dépôt.

Avec les contraintes 3, le nombre d'arcs sortants du dépôt devra être égal au nombre d'arcs entrants. C'est de cette façon que nous créons les circuits.

Les contraintes 4 et 5 permettent d'éviter la création de sous-circuits entre sommets dont l'origine n'est pas le dépôt.

Désormais, nous nous intéresserons uniquement au problème de tournées de véhicules avec capacité. On associe d'une part à chaque véhicule une même capacité Q et d'autre part une certaine demande q_i à chaque client.

Naturellement, un véhicule ne pourra desservir des clients dont la somme des demandes est supérieure à sa capacité. Le modèle décrit ci-haut doit donc prendre en compte cette contrainte de capacité pour supporter l'extension. La contrainte suivante 6 correspond à la contrainte de capacité :

$$\sum_{i=1}^{n}(q_i.\sum_{k=1}^{m}x_{i,j,k}) \leq Q, \quad \forall k \in \{1 \dots m\} \tag{6}$$

2 Le problème de tournées de véhicules par groupes

2.1 Modélisation

Le problème de tournées de véhicules par groupes connu sous le nom de "Clustered Vehicle Routing Problem" hérite des contraintes et variables du problème de tournées de véhicules à capacité. Une nouvelle contrainte vient cependant apparaître pour prendre en compte la notion de groupes.

Un véhicule qui lors d'une tournée rentre dans un groupe (Cluster) doit satisfaire la demande de l'ensemble des clients de ce groupe avant d'en sortir. Il s'agit donc dans ce problème de prendre en compte cette nouvelle contrainte.

Soit C l'ensemble des clusters de la carte de nombre $nbClusters$ et C_u l'ensemble des nœuds appartenant au cluster u. La modélisation du $CCVRP$ partant de la modélisation précédente s'écrit par l'ajout de la contrainte suivante :

$$\sum_{i \in C_u}\sum_{j \notin C_u}\sum_{k=1}^{m}x_{i,j,k} = 1, \quad \forall u \in \{1 \dots nbClusters\} \tag{7}$$

Cette contrainte -combinée à ce qui précède- est suffisante pour exprimer la contrainte du $CCVRP$. On peut aussi identifier certaines contraintes redondantes telles que :

$$\sum_{i \notin C_u}\sum_{j \in C_u}\sum_{k=1}^{m}x_{i,j,k} = 1, \quad \forall u \in \{1 \dots nbClusters\} \tag{8}$$

Mais aussi la combinaison de 7 et de 8

$$\sum_{i \in C_u}\sum_{j \notin C_u}\sum_{k=1}^{m}x_{i,j,k} + \sum_{i \notin C_u}\sum_{j \in C_u}\sum_{k=1}^{m}x_{i,j,k} = 2, \quad \forall u \in \{1 \dots nbClusters\} \tag{9}$$

2.2 Résolution exacte

Il existe une méthode de résolution exacte pour le problème du *CCVRP*. Celle-ci est basée sur le principe de pénalisation de la fonction objectif quand le critère de clustering n'est pas respecté. Cette méthode est très complexe tant sur le plan modélisation que sur le plan temporel.

2.3 Optimisation du problème

Il y a aussi d'autres méthodes non exactes pour l'optimisation de ce problème. Ce sont des métaheuristiques sans garantie d'optimalité.

2.3.1 Méthode du barycentre

La première étape de cette méthode consiste à déterminer un barycentre pour chaque groupe qui sera considéré comme un nouveau client. La résolution du problème de tournées de véhicules sur chacun de ces nouveaux clients permet d'obtenir l'ordre dans lequel les clusters seront visités. Enfin, on déterminera un chemin hamiltonien à l'intérieur de chaque cluster. Cette méthode est notamment décrite dans [ref.3]

2.3.2 Méthode du BIG-M

La méthode du *BIG-M* se rapproche de la méthode à pénalité car elle pénalise les distances entre nœuds de clusters différents en les amplifiant. La valeur du *BIG-M* est suffisamment grande pour éviter autant que possible les arcs entre clusters.

Elles divergent dans le fait que la méthode du *BIG-M* agit directement sur les données en entrée (les distances) et non pas durant la résolution (pénalisation de la fonction objectif). Cette méthode a été traitée notamment dans [ref.4].

3 TSPLIB

3.1 État de la TSPLIB

La TSPLIB est une bibliothèque d'instances pour les problèmes du type TSP (Traveling Salesman Problem), ATSP (Asymmetric Traveling Salesman Problem), CVRP (Capacitated Vehicle Routing Problem), HCP (Hamiltonian Cycle Problem), et SOP (Sequential Ordering Problem). Elle dispose d'un format de fichier standard ASCII permettant de modéliser les instances de ces différents problèmes.

Chaque fichier ASCII est structuré en deux parties distinctes. D'une part, l'en-tête du fichier représente l'ensemble des spécifications. On peut y trouver entre autres le nom du problème, le type auquel il appartient, la dimension du problème (le nombre de sommets du graphe), la capacité des véhicules, le type des coordonnées de chaque sommet du graphe etc ...

D'autre part, l'ensemble des données est composé de sections permettant de représenter les demandes attribuées à chaque sommet du graphe, les coordonnées de chaque sommet du graphe, les distances entre deux sommets du graphe grâce à une matrice des distances.

On peut ainsi modéliser les instances de problèmes de transport en ayant un socle commun et portable.

3.2 Extension de la TSPLIB

M. Sevaux a étendu le format TSLIB [ref.5] pour qu'il puisse supporter les problèmes de tournées de véhicules par groupes. Une nouvelle classe de problèmes a été ajoutée : *CCVRP*.

Cette extension du format introduit essentiellement une nouvelle section dans la partie des données nommée *CLUSTER_SECTION*. Cette section associe à chaque sommet du graphe un cluster. On modélise ainsi l'appartenance d'un ensemble de sommets à un cluster. Trois nouvelles distances ont été introduites :

- *EUC_2D_INT*
- *EUC_2D_DBL*
- *EUC_2D_1DD*

4 VRPH

4.1 Vehicle Routing Problem Heuristics

Vehicle Routing Problem Heuristics (VRPH) est une plateforme logicielle de résolution approchée des problèmes de tournées de véhicules. Cette application a été développée par *Chris Groer* durant sa thèse de doctorat [ref.6].

La plateforme prend en entrée un fichier au format TSPLIB et retourne une solution optimisée. La solution obtenue peut être visualisée sous forme de graphe. Le logiciel n'implémente que la métaheuristique tabou.

Cependant, de nombreuses heuristiques liées au problème de tournées de véhicules et du voyageur de commerce ont été implémentées. Les heuristiques suivantes ont été imlpémentées :

- One point move
- Two point move
- Three point move
- One opt
- Two opt
- OrOpt
- ThreeOpt
- Clarke Wright

Le logiciel VRPH supporte un sous-ensemble du format TSPLIB. Étant un logiciel sur le problème de tournées de véhicules, l'ensemble des options de la TSPLIB permettant de modéliser

les problèmes de tournées de véhicules sont pris en charge. Les instances CVRP de la TSPLIB sont entièrement supportées. Après une lecture du code source du logiciel, on s'aperçoit que le fichier *src/VRPIO.cpp* contient l'ensemble du code permettant le support de la TSPLIB.

Par conséquent, le support d'une extension du format de la TSPLIB par le logiciel VRPH passera nécessairement par la modification de ce fichier. La méthode $read\_TSPLIB\_file$ dont le seul argument est une chaîne de caractère doit notamment attirer notre attention car c'est elle qui est appelée dans tout programme qui prend en entrée un fichier au format TSPLIB.

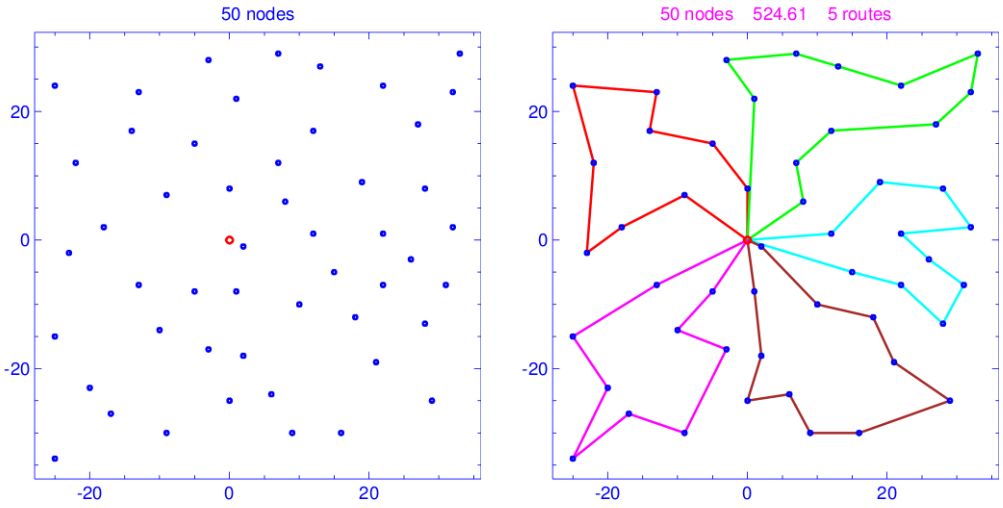


FIGURE 1 – Résolution du problème de tournées de véhicules à capacité

4.2 Traduction de TSPLIB avec distance 2D vers matrice

La première étape du travail a consisté à modifier les instances CCVRP. On transformait ces instances CCVRP en instances CVRP car le logiciel VRPH ne supportait que les instances de ce type.

Pour ce faire, nous avons conçu un logiciel de traduction automatique d'instances. Il prenait en entrée une instance du type CCVRP et générait un fichier CVRP. Le fichier en question contenait une matrice des distances dans laquelle les distances entre sommets de clusters différents étaient multipliées par 10000. C'est de cette façon que l'on introduisait une pénalité.

Enfin, le fichier CVRP généré est transmis au logiciel VRPH. Le logiciel traite donc cette instance comme une instance CVRP classique. Cette démarche pose cependant un certain nombre de problèmes.

On ne peut pas vérifier automatiquement la cohérence du résultat obtenu. Par exemple, nous n'avons aucun moyen de savoir si deux sommets n'appartenant pas au même cluster se situent sur des routes différentes. Il y a une certaine redondance du travail car nous générons une matrice dans un fichier qui est ensuite lu par le logiciel VRPH.

Notons également que la matrice des distances lue par le logiciel VRPH est incorrecte due à l'impact de la pénalité. Ce qui entraîne une valeur incorrecte de la fonction objectif calculée par VRPH. Les problèmes de cette démarche nous invitent à faire des modifications substantielles du code source pour y remédier. On aura ainsi un meilleur contrôle de la résolution des problèmes.

4.3 Prise en compte de TSPLIB CCVRP

Le travail de modification du code source a d'abord consisté à étendre la procédure de lecture de fichier pour supporter le format CCVRP de la TSPLIB. Ce travail s'est concentré essentiellement sur la modification de la méthode $read\_TSPLIB\_file$ du fichier $src/VRPIO.cpp$.

Nous avons ajouté à la classe VRPNode qui modélise un sommet du graphe l'attribut *cluster* permettant de stocker l'identifiant du cluster auquel il appartient. C'est également dans cette méthode que nous calculons la matrice des distances entre deux sommets du graphe. Pour remédier à l'incohérence de la matrice des distances, nous mettons en place deux matrices.

Une première matrice contenant les vrais distances entre deux sommets du graphe. La seconde matrice contient les mêmes distances dont certaines sont pénalisées si elles représentent les distances entre deux sommets de clusters différents.

L'ensemble de la résolution s'effectue sur cette matrice pénalisée. À la fin de la résolution, le calcul de la valeur de la fonction objectif se fera à la fois grâce à la matrice non pénalisée et aux routes obtenues.

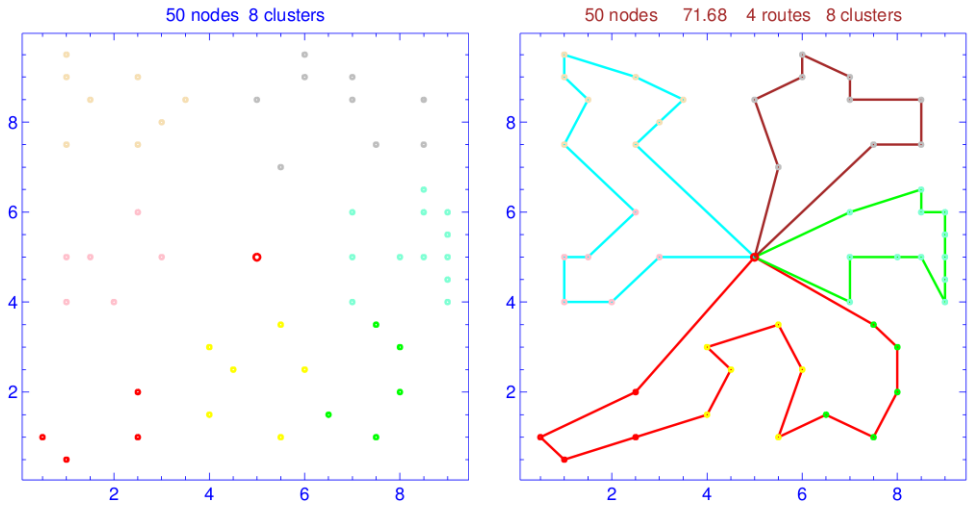


FIGURE 2 – Résolution du Clustered Capacitated Vehicle Routing Problem

5 Valeur du M

La méthode du BIG-M prend comme hypothèse l'amplification de la distance entre les différents clusters selon un paramètre M suffisamment grand.

Une des problématiques rencontrées ici est le choix de sa valeur qui doit être d'une part, assez grande pour éviter de prendre en considération les chemins qui ne respectent pas la contrainte du clustering et d'autre part, ne pas être trop grande pour éviter d'ignorer les vraies distances.

De fait, la représentation machine des nombres réels a tendance à ignorer les petits nombres quand ceux-là sont ajoutés à des nombres très grands. Nous avons envisagé trois manières pour la définition de cette valeur.

5.1 Définition statique

Durant cette étape, la définition de la valeur du BIG-M se faisait de manière statique. On lui attribuait une valeur suffisamment grande. Nous étions vite confrontés aux aléas de cette méthode.

À défaut d'un changement manuel et systématique de la valeur du M pour qu'elle soit adaptée aux données d'entrée, les résultats étaient soit faux soit non optimisés.

5.2 Définition dynamique

Il nous a semblé évident de faire une définition dynamique du BIG-M. Nous avons étudié deux méthodes assez proches l'une de l'autre :

5.2.1 Addition des distances

C'est la première des valeurs calculées dynamiquement. Elle consiste à additionner l'ensemble des distances entre les nœuds. Ce travail n'augmente pas la complexité temporelle car il est fait en même temps que le remplissage de la matrice des distances.

Les premières utilisations de cette méthode étaient cohérentes, mais le principe du clustering n'était pas respecté lorsque le nombre de nœuds était très petit (la somme des distances n'est pas très éloignée de la distance maximale).

Le programme échouait aussi dans la situation où on avait un ensemble de nœuds très proches avec un autre nœud très éloigné d'eux.

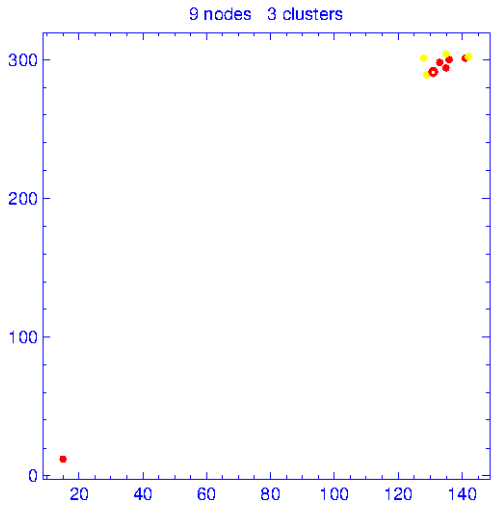


FIGURE 3 – Instance qui ne respecte pas le clustering avec l'addition des distances

5.2.2 Amplification de la distance maximale

La solution mise en place pour remédier aux inconvénients de la méthode ci-dessus, est l'ajout à chaque distance entre nœuds de clusters différents de l'amplification de la distance entre les deux nœuds les plus éloignés de la carte par un facteur multiplicateur. Arbitrairement et après plusieurs essais, ce facteur est pris à 100.

6 Application de la pénalité

La méthode du BIG-M part du principe de pénaliser toutes les arêtes entre deux nœuds qui appartiennent à deux clusters différents. Après avoir défini la valeur de M, il reste à définir la manière avec laquelle l'appliquer. Deux façons sont imaginables dans ce cas :

6.1 Par multiplication

La multiplication de la distance $\delta_{i,j}^{penalized}$ entre les deux nœuds x_i et x_j appartennant à deux clusters différents s'écrirait selon la valeur de M et la distance réelle entre-eux ($\delta_{i,j}^{real}$) de la manière suivante :

$$\delta_{i,j}^{penalized} = M \times \delta_{i,j}^{real}$$

Après le choix de la méthode d'amplification de la distance maximale comme valeur de M, on obtient la fonction suivante :

$$\delta_{i,j}^{penalized} = 100 \times \delta_{max} \times \delta_{i,j}^{real}$$

On remarque que dans le cas d'une distance très petite entre x_i et x_j (inferieure à $2 \,.\, 10^{-2}$) :

$$\delta_{i,j}^{penalized} \leq 2 \times \delta_{max}$$

le problème est encore plus flagrant lorsque la distance entre deux nœuds est inferieure à $1 \, . \, 10^{-2}$ car on se retrouve dans une position où la distance n'est pas dutout pénalisée :

$$\delta_{i,j}^{penalized} \leq \delta_{max}$$

On avait envisagé au cours de cette analyse, d'ajouter une unité à la vraie distance avant de la mutiplier par M, afin d'éviter le cas de figure précédent :

$$\delta_{i,j}^{penalized} = M \times (\delta_{i,j}^{real} + 1)$$

Il reste cependant, un problème qui ne porte pas atteinte à l'intégrité de la résolution, mais qui a une réelle importance dans l'optimisation du problème. Cette méthode favorise les passages entre les clusters les plus proches, malgré le fait qu'ils ne soient pas optimaux.

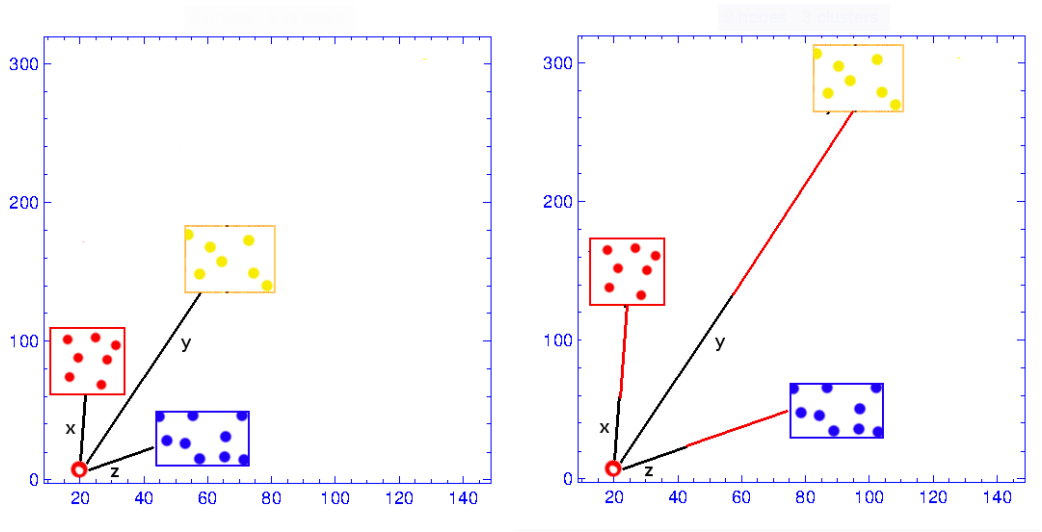


FIGURE 4 – Exemple de différence d'amplification

Une autre configuration où le programme échouait est celle où on avait un ensemble de nœuds très proches, avec un autre nœud très éloigné d'eux.

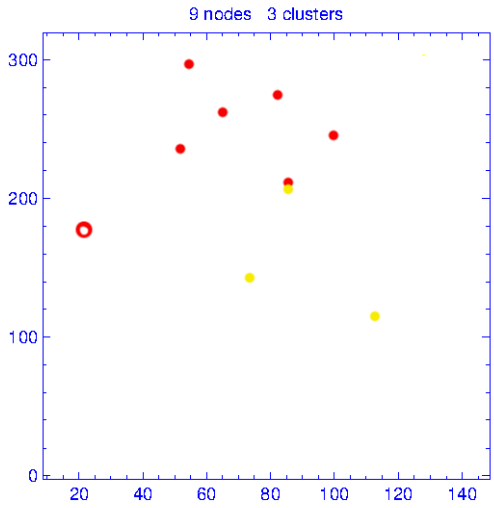


FIGURE 5 – Instance qui ne respecte pas le clustering avec la multiplication du BIG-M

6.2 Par addition

Il s'agit de la méthode choisie au bout de ce TER. La formule de la distance entre deux nœuds x_i et x_j qui n'appartiennent pas au même cluster est définie comme suivant :

$$\delta_{i,j}^{penalized} = 100 \times \delta_{max} \quad + \quad \delta_{i,j}^{real}$$

La méthode de pénalisation s'appuie sur l'ajout d'une distance identique M à chaque fois que le lien du clustering n'est pas satisfait.

Comme indiqué, on ajoute à chaque fois la même valeur. Ceci permet de ne pas avantager un chemin par rapport à un autre. La distinction se fait uniquement selon les vraies distances.

La pénalisation de cette méthode n'est jamais mise à défaut, car hormis dans le cas où toutes les distances entre les différents nœuds sont nulles (un cas inintéressant) :

$$\delta_{i,j}^{penalized} \geq 100 \times \delta_{max}$$

Cette différence (100 fois plus grande) nous permet aisément de distinguer les valeurs pénalisées de celles qui ne le sont pas, afin d'éviter de violer la contrainte d'intégrité.

On note aussi un cas peu fréquant mais possible, où la vraie distance est tellement marginale qu'elle est ignorée. Ceci arrive dans le cas où elle est énormément faible par rapport à la distance maximale.

L'apparition de ce cas de figure dépend de l'architecture sur laquelle on se trouve. Plus la représentation machine des réels est précise, moins en a de chances de tomber dans ce cas. Il faut donc favoriser une architecture $64bit$ plutôt qu'une $32bit$.

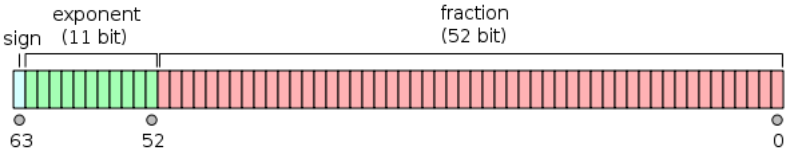


FIGURE 6 – Représentation machine des doubles sous le format IEEE 754, Source Wikipedia

7 Instances de test

Les instances considérées sont des instances de laboratoire. La distance prise en compte pour chaque couple de nœuds représentés par leurs positions dans l'espace est la distance Euclidienne.

La construction de VRPH permet la prise en compte d'une matrice fixe des distances. Nous avons agi en conséquence, afin de garder cette particularité lors de l'extension de l'application au clustering. On aurait pu donc utiliser des instances plus proches de la réalité.

7.1 Mises à disposition

Les premiers essais ont été réalisés grâce à des instances mises à dispositions par *M.SEVAUX*. Ces instances étaient de taille variable, tant sur le nombre de nœuds (de 9 à 199) que sur le nombre de clusters (de 2 à 16).

Ces instances nous ont bien servi pour appréhender le problème. Elles nous ont montré une des façons de le représenter et de distinguer ses similitudes et ses différences avec un problème *CVRP*.

Elles nous ont aussi permis de vérifier le bon déroulement de la résolution de VRPH, mais aussi de corriger les différents *bugs* de l'application.

Le point essentiel fut la confirmation de ce que nous attendions d'un point de vue téorique sur le choix de la valeur du BIG-M, ainsi que de la manière avec laquelle on doit l'impacter.

7.2 Génération d'instances

Au fur et à mesure que le projet avançait, nous avons eu besoin d'étendre nos tests pour prendre en compte des cas spéciaux, afin de voir la réaction de notre application dans différentes situations.

7.2.1 Manuelle

Cette façon de faire a été bien utile pour cibler des cas de figures très spécifiques tels qu'une localisation de l'ensemble des nœuds sur une zone très dense avec des distances très faibles entre les nœuds, ou bien des cas où on a des nœuds marginaux et très éloignés des autres.

Cette méthode a été la moins utilisée du fait de la difficulté de placement des nœuds sur la carte, ainsi que l'attribution des capacités selon le cluster auquel appartient chacun d'entre-eux, tout en garantissant l'intégrité de l'instance.

7.2.2 Automatique

Nous avons mis en place un programme de génération d'instances de type *TSPLIB*. L'instance est générée de manière aléatoire avec un nombre de nœuds et de clusters donnés en paramètres. Le programme prend soin d'affecter à chaque nœud :

- une capacité qui doit être moins élevée que la capacité donnée en paramètre.
- un cluster de façon à éviter leur chevauchement et de se retrouver à la fin avec un cluster par zone bien distincte.

La fin de l'exécution du programme s'effectue par la désignation de la capacité globale d'un véhicule. Celle-ci doit être comprise entre la capacité du cluster de plus grande capacité et la somme des capacités de l'ensemble des nœuds :

$$max\{\ capacity_{cl_k}\ \mid\ cl_k\ \in\ Clusters\ \}\quad \leq\quad capacity_{vehicule}$$

$$capacity_{vehicule}\quad \leq\quad \sum_{i=1}^{n} capacity_i$$

Les instances obtenues à la fin de cette étape avaient la garantie de respecter la définition des instances CCVRP ainsi que le format étendu de la *TSPLIB*.

8 Résultats théoriques

8.1 Faiblesse des métaheuristiques

On s'est appuié pour l'optimisation des problèmes sur un ensemble de métaheuristiques sans gage d'optimalité de la solution retournée. L'application VRPH utilise un algorithme *TABOU* (dont les résultats théoriques montrent qu'il tend vers la solution optimale quand le nombre de tours tend vers l'infini) pour la résolution du problème. Il passe par des solutions intermédiaires qui ne respectent pas forcément le principe du clustering.

En se basant sur les deux remarques ci-dessus et en considérant que le temps d'exécution n'est pas infini, on sait clairement qu'il y'aura des cas qui ne seront pas cohérents avec le problème, du fait du temps d'exécution.

8.2 Favoritisme

On s'est rendu compte lors de notre étude théorique que certaines situations étaient plus favorisées que d'autres. Voici un exemple :

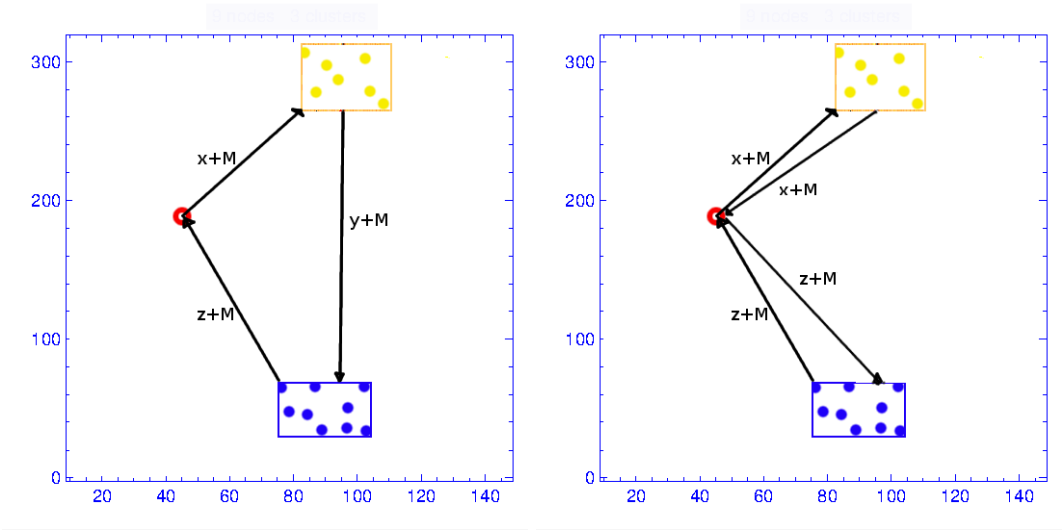


FIGURE 7 – Exemple de favoritisme

Dans l'exemple précédent, la distance entre les clusters du premier schéma est de :

$$x + M \ + \ y + M \ + \ z + M \ = \ x + y + z \ + \ 3M$$

alors que celle du deuxème schéma est de :

$$x + M \ + \ x + M \ + \ z + M \ + \ z + M \ = \ 2x + 2z \ + \ 4M$$

Le deuxième cas est alors automatiquement ignoré. Cette situation ne pose pas de problème dans le cas de distances Euclidiennes (sauf dans le cas où les deux clusters et le dépot sont sur la même droite).

Le problème est flagrant lorsqu'il s'agit de distances *réelles* indiquées de manière fixe. La méthode du *BIG-M* peut exclure la solution optimale au profit d'une autre qui ne le serait pas.

8.3 Utilisation du Clarke & Wright

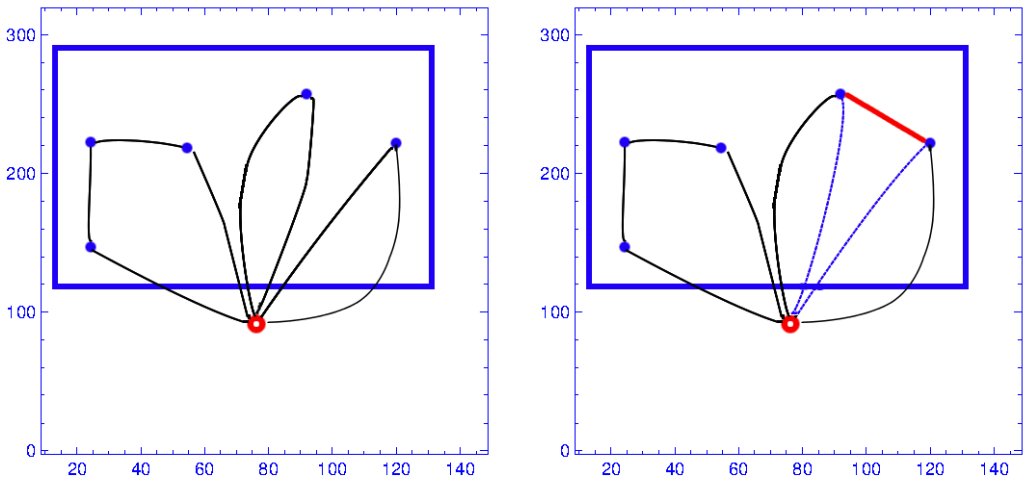
Nous avons essayé de voir si la méthode du *BIG-M* arrivait à une solution cohérente vis-à-vis de la contrainte du clustering à la fin de son exécution (plus d'améliorations possibles).

Normalement, un arc entre deux nœuds appartennant au même cluster devrait être favorisé par rapport à deux autres qui n'y appartiennent pas. À chaque itération, on devrait retirer un arc entre un nœud et le dépot pour le remplacer par un autre entre deux nœuds. Deux possiblités dans ce cas :

- Les nœuds appartiennent au même cluster.
- Les nœuds appartiennent à deux clusters différents.

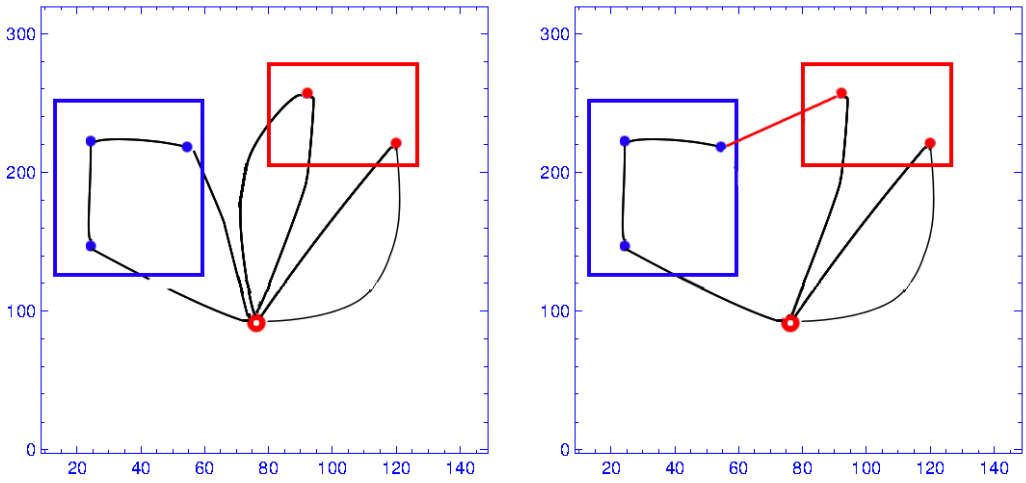
Appartiennent au même cluster

Voici un exemple d'un remplacement d'arc dans ce cas :



Appartiennent à deux clusters différents

Voici un exemple de remplacement de deux arcs dans chacun relie le dépot à un nœud qui n'appartient pas au même cluster :

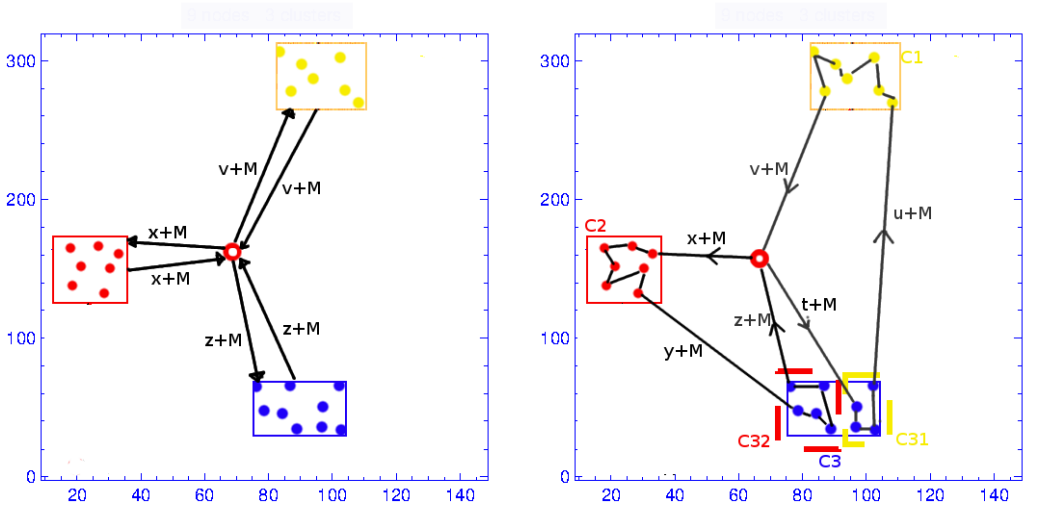


Cas d'une possible erreur

Nous avons pu identifier au cours de ce travail, un cas de figure qui peut poser problème. Il s'agit du cas où les nœuds d'un cluster peuvent être distribués sur deux routes différentes sans que cela ne viole la contrainte sur la capacité des véhicules.

Dans ce qui suit, un exemple où le véhicule est de capacité 30. Les clusters C_1, C_2, et C_3 ont une demande de 20 chacun. Le cluster C_3 peut être décomposé en deux sous-clusters C_{31} et C_{32} dont la demande de chacun d'entre-eux est de 10.

Dans ce cas, deux possiblités s'offrent à nous :



La première configuration est celle souhaitée. Elle cumule (sans addition des distances à l'interieur de chaque cluster) un total de :

$$2(x+M) \ + \ 2(v+M) \ + \ 2(z+M) \quad = \quad 2x+2v+2z \ + \ 6M$$

La deuxième est celle qui risque d'arriver et qu'on cherche à éviter. Elle cumule une distance de :

$$(x+M \ + \ y+M \ + \ z+M) \ + \ (t+M \ + \ u+M \ + \ v+M) \quad = \quad x+y+z+t+u+v \ + \ 6M$$

On remarque que dans ce cas de figure, la différence entre les deux situations ne se fait plus sur le *BIG-M*, mais uniquement sur les vraies distances. Ceci pose un énorme problème dans le cas des distances non Euclidiennes.

9 Intégrité des résultats

Sans preuve mathématique sur la complétude du programme, il faut en moins avoir un moyen qui nous permette de confirmer ou d'infirmer la cohérence des résultats.

Vérification visuelle

9.0.1 Par positionnement

Elle nous oblige d'une part, à identifer chacun des nœuds avec sa position dans l'espace, à partir de la position indiquée dans le fichier de l'instance en entrée. D'autre part, à vérifier que les navettes en sortie satisfont bien la contrainte de clustering.

Cette tâche est encore plus ardue lorsqu'on a beaucoup de nœuds et dans un espace très dense. Ceci rend cette méthode la moins fonctionnelle et la plus subjecte à des erreurs. Voici un exemple d'affichage obtenu à cette étape.

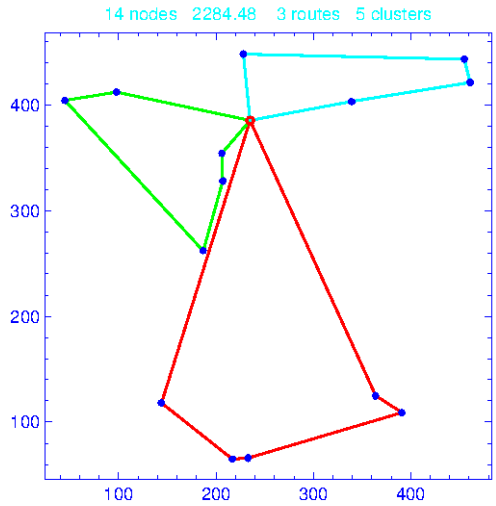


FIGURE 8 – Exemple de l'affichage initial

9.0.2 Par coloration

Nous avons pu mettre à profil la capacité de l'application graphique utilisée (*plplot*) pour identifier les clusters par des couleurs différentes. Ainsi, il suffit de vérifier que l'ensemble des nœuds d'une même couleur apparaissent de manière successive. Évidemment, celà implique aussi qu'ils doivent appartenir à la même tournée de véhicule.

On a été confronté au cours de ces tests à la contrainte technique du nombre de couleurs représentables par l'application. De fait, elle ne permet que la représentation de 16 couleurs distinctes. Il a donc fallu trouver une autre solution pour les cas où le nombre de clusters serait plus élevé.

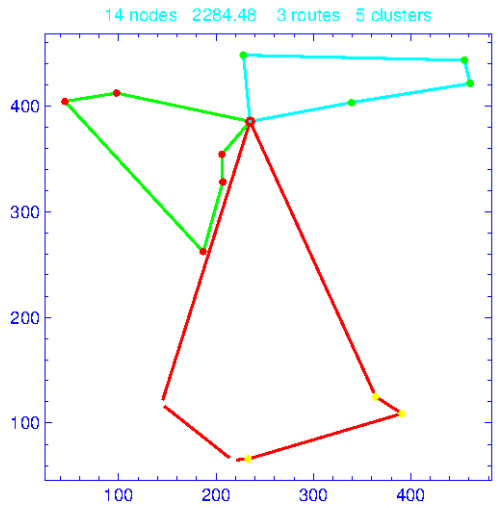


FIGURE 9 – Exemple de l'affichage avec coloration

9.0.3 Par encadrement

Le choix du changement de l'interface graphique (passage de *plplot* vers *gnuplot*) fut très bénéfique. Outre la simplicité d'utilisation, cette application permet l'encadrement des différents nœuds.

Il faut donc vérifier que l'ensemble des nœuds qui composent chaque cluster et qui sont encadrés par un rectangle soient reliés par un unique chemin. Il devient donc assez facile de voir le résultat et de pouvoir juger de sa cohérence et de son intégrité en même temps.

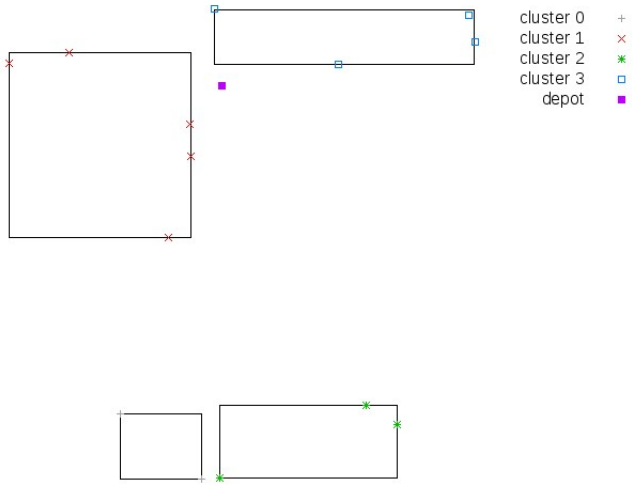


FIGURE 10 – Exemple de l'affichage avec encadrement

Vérification automatique

Nous avons voulu étendre l'application pour qu'elle puisse elle-même, nous confirmer la cohérence du résultat avant de nous l'afficher. Ceci, afin d'éviter toute ambiguïté sur la nature du résultat (ne pas retourner le résultat s'il est erroné).

Nous avons mis en place un algorithme de test qui parcourt l'ensemble des tournées. Pour chaque nœuds appartennant à cette tournée (hormis le dépot), s'il n'est pas du même cluster que celui qui le précède et que son cluster a déjà été visité, ceci lève une exception pour signaler le cas d'erreur.

Cet algorithme n'aurait pas de sens dans le cas où un des cluster possède une capacité plus élevée que les véhicules de la flotte. Ce cas est évidemment évité directement dans la création des instances, car il ne satisfait pas les règles du *CCVRP*.

Cette méthode nous donne la possibilité d'accélérer la résolution des instances tout en ayant la certitude sur la cohérence des résultats. Voici l'algorithme mis en place pour la vérification de la cohérence :

```
Consistency( listOfRoutes: List<route>): Boolean
Begin:
|  visited[numberOfClusters] # intialised with False
|  oldCluster <= -1
|  for each route in listOfRoutes
|  do
|  |  for each node in route
|  |  do
|  |  |  if( node = depot ) then
|  |  |  |  oldCluster <= depot.cluster
|  |  |  else:
|  |  |  |  if ( node.cluster <> oldCluster and visited[node.cluster] = True) then
|  |  |  |  |  return False
|  |  |  |  else
|  |  |  |  |  oldCluster <= node.cluster
|  |  |  |  |  visited[node.cluster] <= True
|  |  |  |  fi
|  |  |  fi
|  |  done
|  done
|  return True
end.
```

10 Génération et test automatique d'instances

Pour avoir un aperçu des résultats que l'application retourne, il nous a semblé utile de mettre en place une routine de génération et de test. Pour cela, nous avons mis en place un script qui combine les trois composants :

- Génération automatique d'instances *CCVRP*.
- Optimisation du problème avec la nouvelle extension de VRPH.
- Vérification de l'intégrité du résultat obtenu.

Nous avons fait tourner ce script toute une journée de manière automatique et de façon à prendre en compte plusieurs paramètres. Ces paramètres servent à distinguer l'instance qui devait être créée à chaque fois. Ainsi, nous avons pu générer des jeux de test avec un nombre de nœuds allant de 2 à 2000 et avec un nombre de clusters de 1 à 50.

Cette étape ne se limite pas uniquement à cela, car on a permis la génération de plusieurs instances pour les mêmes paramètres (génération de vingt instances dans notre cas).

Nous avons eu au final tout un répertoire d'instances *CCVRP*, ainsi qu'un autre qui contient la meilleure solution obtenue pour chacune d'entre-elles.

Il est à noter que nous n'avons eu aucun échec sur l'ensemble de l'exécution. Ceci n'est pas une garantie en soi pour confirmer que le programme est complet, mais il conforte au moins la position de l'application et de l'heuristique (*BIG-M*) pour la résolution de tels problèmes.

11 Organisation du TER

La réussite du TER dépend fortement du mode d'organisation sur lequel il s'appuie. On a mis en place en accord avec M.SEVAUX, un plan et un processus d'étude, mise en place (de développement) et de *reporting* de l'avancée du travail.

11.1 Réunions par Visio-conférence

Du fait de l'éloignement géographique avec notre encadrant, nous avons utilisé le moyen de contact mis en place par le laboratoire qui est la salle de visio-conférence. On a prévu une réunion de ce type toutes les deux semaines. La fréquence a un peu baissé à la fin du TER du fait de la clarification des objectifs.

Nous avons eu au départ quelques soucis d'adaptation à ce mode de fonctionnement. Nous n'avions pas l'habitude et le réflexe des réservations de salles. Ce qui a rendu cette tâche parmi les plus ardues. Il fallait prendre contact avec les différentes personnes concernées et bien sûr, veiller à ce que notre encadrant soit libre au même moment (avec une contrainte sur la disponibilité des salles des deux cotés).

Les réunions étaient jumelées avec un autre groupe (celui dont le projet porte sur le thème de la parallélisation du CCVR). Ceci a été très bénéfique pour nous, car il nous a permis d'avoir des idées et des notions utiles pour une future parallélisation de VRPH (exemple de la parallélisation de l'application des métaheuristiques et des calcules effectués à l'interieur de chacune d'entre-elles).

Nous avons appuyé ces réunions par des points d'avancement sur les objectis fixés. Cela se faisait par mail. Nous avons mis comme obligation de le faire au minimum une fois par semaine. Il nous a permis de ne pas diverger de l'objectif. Il nous a aussi servi comme base pour la préparation des réunions qui suivaient.

11.2 Comptes rendus des réunions

Au cours d'une réunion, un rédacteur est désigné pour faire une sorte de procès-verbal de la réunion. Le premier d'entre-eux est joint en annexe. Celui-ci devait mentionner les personnes présentes à la réunion, un résumé du travail effectué et les informations et conseils de l' encadrant.

À la fin d'une réunion, on se donne un objectif à atteindre. Cet objectif doit être clairement explicité dans ce compte rendu.

Les dernières réunions ont moins porté sur notre sujet. L'objectif était le même (le développement de l'application et le test de cohérence des résultats). Les autres points (présence et résumé du travaille effectué) étaient inclus dans le compte rendu du deuxième groupe.

11.3 Recherche de sources d'information

11.3.1 Lecture d'articles

Comme le montre les différentes références, nous nous sommes appuyés au cours de notre travail de recherche sur plusieurs articles et livres. Certains traitaient des problèmes que nous abordons dans ce rapport et d'autres de sujets annexes.

L'ensemble de la bibliographie n'a pas été citée, seules les ressources indispensables pour appuyer nos dires ont été référencées dans la bibliographie.

11.3.2 Échanges

Évidemment, la majorité des interrogations ont été eues avec notre encadrant. Son experience fut la plus bénéfique. On a aussi eu des discutions avec *Thibaut Barthélémy*, qui a participé au développement de cette méthode (du *BIG-M*) lors de son stage réalisé l'an dernier.

Il nous a donc mis à disposition les instances sur lequelles sa métaheuristique échouait (algorithme de recherche tabou). Il nous a aussi exposé son explication sur cet échec et les raisons qu'il a pu identifier ainsi qu'un moyen pour y remédier (utilisation de deux valeurs différentes pour le *BIG-M*, une entre les différent clusters et une autre entre un cluster et le dépot).

11.4 Une version VRPH par amélioration

Afin de ne pas perdre nos différentes étapes de développement et à défaut d'avoir un gestionnaire de version de type *SVN* ou *GitHub*, nous avons décidé de faire une version pour chaque mise à jour (modification ou/et ajout) du code. Le passage d'une version à une autre devait être indiqué avec une signalisation claire.

11.4.1 Identification d'une version

Une version contient deux fichiers qui servent à informer de ce que peut ou ne peut pas faire le programme.

11.4.1.1 Tâches faites (@RELEASE)

Le fichier *RELEASE* est le fichier à lire avant toute utilisation. Nous avons opté pour y répertorier un bref résumé de l'application, une liste des modifications et des améliorations apportées dessus, ainsi que les remarques négatives et cas qui posaient problème à ce stade.

11.4.1.2 Tâches à faire (@TODO)

Le fichier *TODO* est une sorte d'objectif à atteindre lors de la prochaine version. Il propose des solutions techniques pour remédier à un ou plusieurs problèmes indiqués dans le *RELEASE*.

11.4.2 Passage aux versions superieures

On a eu au cours de ce TER, huit versions plus au moins importantes les unes des autres. Toutes les versions ne méritaient pas d'être transmises à notre encadrant. Nous avons jugé utile de lui envoyer trois d'entre-elles, afin d'avoir un avis d'expert sur le travail déjà effectué :

- Transformation externe des instances *CCVRP* vers *CVRP*.
- Modification interne de VRPH pour le *CCVRP*.
- Version finale avec correction de l'ensemble des anomalies identifiées.

12 Les acquis

Au cours de ce semestre, nous avons été en pleine immersion dans un sujet scientifique nouveau dont les résultats ont un grand intérêt pour les enterprises du domaine.

Le *CCVRP* est un cas spécial du *CVRP* qui est un des problèmes les plus répendus (problèmes de logistique). On a pu étudier de près les différents moyens mis en œuvre pour la résolution du problème de tournées de véhicules (l'ensemble des métaheuristiques appliquées). Ce pas était crucial pour la prise en main de *VRPH*. Il nous a donc permis d'avoir un pas d'avance sur le module *Logistique* initié en *Master 2 ORO*.

Le sujet qu'on a eu à traiter offre l'avantage d'avoir été traité et d'avoir encore à ce jour des améliorations et optimisations possibles. Il nous a donc permis d'avoir un aperçu des travaux déjà effectués et nous a laissé l'espace qu'il faut pour imaginer et mettre en place de nouvelles idées.

Comme indiqué ci-dessus, l'existence d'articles ayant déjà traité ce sujet nous a permis de faire un travail d'investigation et de recherche. Il nous a aussi permis de savoir orienter notre recherche de sources d'information et de n'en garder que le plus intéressant.

Le caractère applicatif du projet (extension d'une application) nous a permis d'approfondir nos connaissances en génie du logiciel tant du côté modélisation (avec le *refactoring* du code existant pour sa compréhension) que du côté développement (utilisation des librairies *C++*, *plplot* et *gnuplot*).

Nous avons pu être confrontés au mode de fonctionnement d'un laboratoire où les outils sont partagés et demandent à être réservés (en particulier, la salle de vision-conférence).

Nous avons vu de plus près, les étapes de réalisation d'un projet de recherche. Nous avons participé à la mise en place du processus.

13 Améliorations

L'application mise en place est d'un grand intérêt pour l'étude du problème. Malgré cela, il faudra passer par une preuve mathématique, afin de confirmer ou d'infirmer la cohérence des résultats.

La preuve peut être dans le cas d'une résolution exacte. Ceci lèvera définitivement le doute sur la *consistance* de la méthode du *BIG-M*.

On peut aussi chercher à trouver les conditions suffissantes pour la cohérence des résultats lors d'une optimisation par métaheuristiques.

En plus de la cohérence des résulats, il faudra aussi mesurer l'impact de l'amplification des distances entre clusters par la valeur du *BIG-M*. Ainsi, il faudra voir s'il n'y a pas de solutions disciminées, alors qu'elles n'auraient pas dû l'être.

On peut également, imaginer d'autres types d'instances à tester avec l'application mise en place. Cela ne va pas faire office de preuve, mais permet en moins d'étendre le champ se son d'utilisation

Conclusion

Pour conclure, nous avons pu aborder de nombreux aspects de la recherche. Ceci nous a conforté dans notre envie d'en faire et de cotoyer ainsi des problèmes auxquels l'homme fait face tous les jours.

Le caractère à la fois théorique et pratique de ce sujet de recherche nous a permis de s'y imprégner plus facilement. On a pu constater les dialogues incessants entre le travail théorique et pratique, l'un n'allant pas sans l'autre.

Les nombreux articles que nous avons pu consulter nous ont donné un aperçu de la rigueur appliquée au cours d'une recherche scientifique. Nous sortons satisfaits de ce travail car plus que jamais l'acquis de connaissance reste quelque chose d'inestimatible.

Références

[ref.1] Cattrysse, Sevaux, Sörensen, Van den Bergh. *A multi-objective distribution problem for parcel delivery at TNT.*

[ref.2] Ben Ismail, Legras, Coppin. *Synthèse du problème de routage de véhicules.*

[ref.3] Sevaux, Sörensen. *Hamiltonian paths in large clustered routing problems.*

[ref.4] Barthélémy, Rossi, Sevaux, Sörensen. *Metaheuristic Approach for the clustered VRP.*

[ref.5] Sevaux. *Clustered Capacitated Vehicle Routing File Format.*

[ref.6] Groer. *Parallel and serial algorithms for vehicle routing problems.*2008